Chapter

5

Optimization of Unconstrained Multivariable Functions

5.1 Introduction

When the optimization of an objective function is required without any additional correlation, then this optimization is called unconstrained optimization. Unconstrained optimization problem appears in some cases in chemical engineering. It is the simplest multivariable optimization problem. Parameter estimation is a significant application in engineering and science, where, multivariable unconstrained optimization methods are required. Some optimization problems are inherently unconstrained; there is no additional function (section 2.7, 2.8). When there are some usual constraints on the variables, it is better to ignore these constraints and to consider that they do not have any impact on the optimal solution. Unconstrained problems also formed due to reconstructions of constrained optimization problems, in which the penalization terms are used to replace the constraints in the objective function that have the effect of discouraging constraint violations.

Rarely do we get any unconstrained problem as a practical design problem, the knowledge on this type of optimization problems is essential for the following purposes:

1. The constraints hold very less influence in some design problems.
2. To get basic idea about constrained optimization, the study of unconstrained optimization techniques is necessary.
3. Solving the unconstrained optimization problem is quite easy compared to constrained optimization.

Some robust and efficient methods are required for the numerical optimization of any nonlinear multivariable objective functions. The efficiency of the algorithm is very significant because these optimization problems require an iterative solution process, and this trial and error becomes unfeasible when number of variables is more than three. Generally, it is very difficult to predict the

behavior of nonlinear function; there may exist local minima or maxima, saddle points, regions of convexity, and concavity. Therefore, robustness (the capability to get a desired solution) is desirable for these methods. In some regions, the optimization algorithm may proceed quite slowly toward the optimum, demanding excessive computational time. In this chapter, we will discuss various nonlinear programming algorithms for unconstrained optimization.

5.2 Formulation of Unconstrained Optimization

For an unconstrained optimization problem, we minimize the objective function that is constructed with real variables, without any limitations on the values of these variables. The mathematical representation of this minimization problem is:

$$\min_{x} f(x) \tag{5.1}$$

where $x \in \mathbb{R}^n$ is a real vector with $n \geq 1$ components and $f : \mathbb{R}^n \to \mathbb{R}$ is a smooth function.

Generally, it is desirable to find the global minimum of $f(x)$, a point where the objective function reaches its least value. We may face difficulty to find the global minima, since our knowledge on $f(x)$ is typically local only. We do not have an adequate idea on the overall shape of $f(x)$ as our algorithm does not visit thoroughly many points. Therefore, we can never be confirmed whether the function does not have a sharp valley or peak in some area that is not examined thoroughly by the algorithm. The majority optimization algorithms are capable of finding only a local minimum. Local minimum is a point that produces the least value of $f(x)$ in neighborhood of the starting point. The methods discussed in the following sections are local minimization method.

5.3 Direct Search Method

Hook and Jeeves coined the term "direct search" in 1961 [Hooke and Jeeves, (1961)]. Sometimes the classical methods fail or are not feasible to find the maximum or minimum value of a complicated function. 'Direct search' is a method for solving these problems by using a computer that follows a simple search strategy [Hooke and Jeeves, (1961)]. The phrase "direct search" has been used to describe sequential analysis of trial solutions by performing the comparison of each trial solution with the "best" obtained up to that time. This method also finds a strategy for determining what the next trial solution will be (as a function of earlier results) [Lewis *et al.* (2000)]. Although, most direct search methods have been developed by heuristic approaches, some of them have proved extremely effective in practice, particularly in applications where the objective function was non-differentiable, had discontinuous first derivatives, or was subject to random error [Swann (1972)].

5.3.1 Random search methods

A large class of optimization problems can be handled by random search techniques. Random search techniques were first introduced by Anderson [Anderson, (1953)] and later by Rastrigin [Rastrigin, (1963)] and Karnopp [Karnopp, (1963)]. The particular type of problem is presented here is called an optimization problem and is characterized by a search process, carried out in a

multidimensional space, $X = x_1, x_2, \ldots, x_n$. The search process is conducted for a set of values of the parameters (X) that gives an absolute extreme (minimum or maximum) of a criterion or reward function,

$$f(X) = f(x_1, x_2, \ldots, x_n) \tag{5.2}$$

Random numbers are utilized in these methods for determining the minimum point. These methods can be used quite efficiently as random number generator is available with most of the computer libraries. Here, we have presented some of the well-known random search methods.

5.3.1.1 Random jumping method

This method randomly generates points $x^{(k)}$ within a fixed region; selecting the one that provides the best value of the function over a huge number of trials. The Fig. 5.1 represents the random jumping method. The contour is given for an objective function. Minimum value of the function has been found using random jumping method. The algorithm of this method is given below:

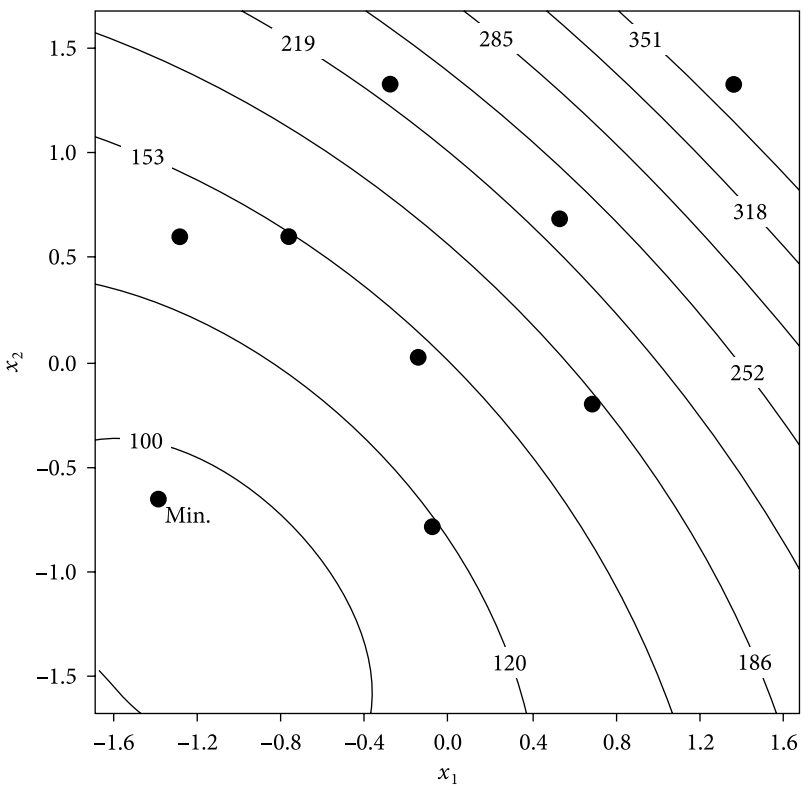

Fig. 5.1 Contour representation for random jumping method

Algorithm

Step 1 Formulate the optimization problem with a lower (l_i) and upper (u_i) bounds for each design variable x_i, $i = 1, 2, \ldots, n$ to generate the random values of x_i.
where $l_i \le x_i < u_i$, $i = 1, 2, \ldots, n$

Step 2 Sets of n random numbers ($r_1, r_2, \ldots, r_n$) were generated. These random numbers are distributed uniformly between 0 and 1. Each set of these numbers is utilized to locate a point X, which is defined as

$$X = \begin{Bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{Bmatrix} = \begin{Bmatrix} l_1 + r_1(u_1 - l_1) \\ l_2 + r_2(u_2 - l_2) \\ \vdots \\ l_n + r_n(u_n - l_n) \end{Bmatrix} \tag{5.3}$$

At point X, the value of the function is calculated.

Step 3 Generate a huge number of random points X and evaluate the value of objective function at each of these points.

Step 4 For a minimization problem, select the smallest value of $f(X)$ as it is preferred. Conversely, select the highest value of $f(X)$ for a maximization problem.

5.3.1.2 Random walk method

Random Walk Optimization (RWO) is an optimization process, which is a zero order method. Zero order implies that no derivatives, only the function values are used to found the search vector. This method is suitable for non-smooth, discrete objective functions as it does not require the derivative of the functions. During each iteration, the direction of search process is a random direction. Random Walk optimization algorithm is a very simple and therefore, the optimization process is controlled by very few parameters.

The random walk method works on the principle of generating a series of approximations that improves gradually toward the minimum. Each of these approximations is determined from the previous approximation. Therefore, if X_i is the approximation to the minimum attained in the $(i - 1)$th iteration (or stage or step), the improved or new approximation in the ith iteration is estimated by the relation

$$X_{i+1} = X_i + \lambda u_i \tag{5.4}$$

where λ is a specified scalar step length and u_i denote a unit random vector generated in the ith iteration. The following steps [Fox, (1971)] describe the detailed operation of this method:

1. Start with the following parameters: X_1, initial point; an adequately large initial step size of λ; ε, a minimum permissible step size, and N, the maximum allowable number of iterations.
2. Calculate the value of the function $f_1 = f(X_1)$.
3. Set the iteration number as $i = 1$.

4. Create a set of n random numbers $r_1, r_2, \ldots, r_n$; all are enclosed in the interval [–1, 1] and express the unit vector u as

$$u = \frac{1}{\left(r_1^2 + r_2^2 + \cdots + r_n^2\right)^{1/2}} \begin{Bmatrix} r_1 \\ r_2 \\ \vdots \\ r_n \end{Bmatrix} \tag{5.5}$$

It is expected that the directions created using Eq. (5.5) have a bias toward the diagonals of the unit hypercube [Fox, (1971)]. To keep away from such a bias, the length of the vector (R) is calculated as

$$R = \left(r_1^2 + r_2^2 + \cdots + r_n^2\right)^{1/2} \tag{5.6}$$

and the acceptance of the random numbers generated ($r_1, r_2, \ldots, r_n$) depend on the value of the vector (R). They are accepted only when $R \leq 1$ but are rejected if $R > 1$. When we accept the random numbers, then the unbiased random vector u_i is represented by Eq. (5.5).

5. Calculate the new vector as $X = X_1 + \lambda u$ and the corresponding value of the function $f = f(X)$.
6. Then the values of f_1 and f are compared. When $f_1 > f$, set the new values as $X_1 = X$ and $f_1 = f$ and go to step 3. If $f_1 \leq f$, move to step 7.
7. If $i \leq N$, update the new iteration number as $i = i + 1$ and go to step 4. Conversely, if $i > N$, move to step 8.
8. Calculate the reduced new step length as $\lambda = \lambda/2$. Whenever the new step length $\lambda \leq \varepsilon$, move to step 9; if not (i.e., if the new step length $\lambda > \varepsilon$), go to step 4.
9. Terminate the process by considering $X_{opt} = X_1$ and $f_{opt} = f_1$.

5.3.2 Grid search method

Grid search process includes mainly three steps: constructing a proper grid within the design space, estimating values of the objective function at all these grid points, and finally locating the grid point with the lowest function value (highest function value for maximization problem).

Consider for the ith design variable, the lower and upper bounds are represented by l_i and u_i, respectively. Split the entire range [l_i, u_i] into $p_i - 1$ equal parts so that $x_i^{(1)}, x_i^{(2)}, \ldots x_i^{(p_i)}$ indicate the grid points along the x_i axis ($i = 1, 2, \ldots, n$), where n is the number of variables. This gives us a total $p_1 \times p_2 \times \ldots \times p_n$ number of grid points within the design space. Figure 5.2 shows a grid with $p_i = 7$ for a two-dimensional design space.

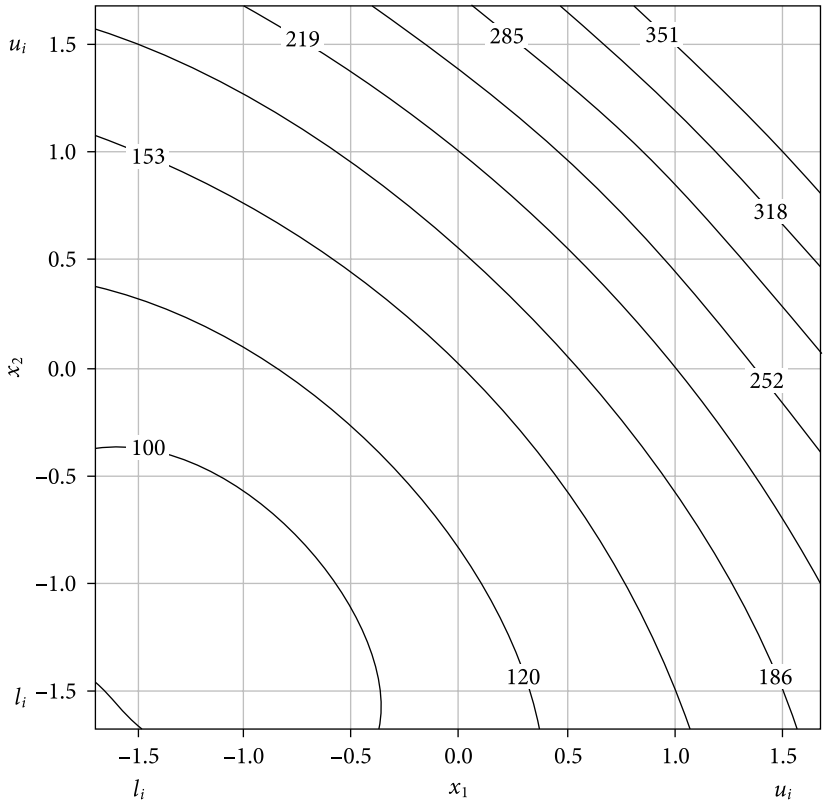

Fig. 5.2 Contour representation for grid search method

We can select the grid points based on methods of experimental design [Montgomery, (2008)]. It can be noticed that in most practical problems the grid search method needs a huge amount of function evaluations. For instance, consider a problem having 8 design variables (n = 8), the number of grid points will be 4^8 = 65536 with p_i = 4 and 5^8 = 390625 with p_i = 5. However, the grid search method can be applied easily to locate an approximate minimum for problems that have small number of design variables. In addition, the grid search method can be employed to establish an appropriate starting point for other more effective methods such as Newton's method. Details of this method with examples have been discussed in chapter 11.

Example 5.1

Experimental results show that the Cr(VI) removal by Powder Activated Carbon (PAC) at pH 4, follows the relation

$$\text{Cr(VI) removal(\%)} = 77.92 + 9.41C + 3.86t - 3.53C^2 - 7.33t^2 \quad \text{[Dutta } et\ al. \text{ (2011)]}$$

Find the optimum value of time and PAC concentration which will show maximum Cr(VI) removal.

Here, we solve the problem using grid search method. This is a two dimensional problem (time and PAC concentration as shown in Fig. 5.3) with upper limit and lower limit as follows:

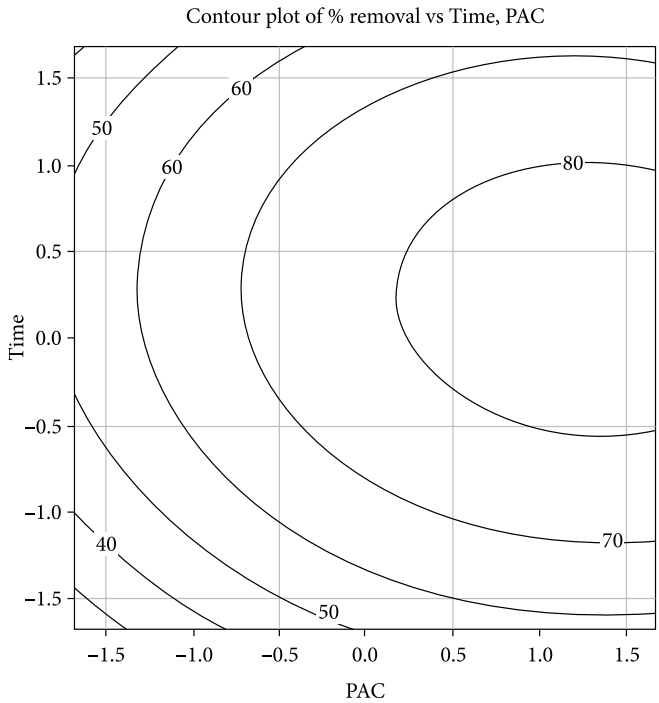

Fig. 5.3 Contour representation for Cr(VI) removal

Say PAC concentration (C) is denoted as variable 1 and time (t) is variable 2;

$l_1 = -1.5$, $u_1 = 1.5$ (Coded value)

$l_2 = -1.5$, $u_2 = 1.5$ (Coded value)

here, we consider $p_1 = 4$, $p_2 = 4$; therefore, we have $p_1 \times p_2 = 4 \times 4 = 16$ grid points.

from the contour plot it is clear that we have to find the function values of these 16 points.

The value of grid points are given in the Table 5.1. Study of various grid points show that the maximum function value is found at (1.5,0.5) point. The corresponding function value is 84.19

Table 5.1 Value of objective function at different grid points

Grid point	Cr(VI) removal (%)	Grid point	Cr(VI) removal (%)	Grid point	Cr(VI) removal (%)	Grid point	Cr(VI) removal (%)
(–1.5, 1.5)	45.16	(–0.5, 1.5)	61.63	(0.5, 1.5)	71.04	(1.5, 1.5)	73.49
(–1.5, 0.5)	55.96	(–0.5, 0.5)	72.43	(0.5, 0.5)	81.84	(1.5, 0.5)	84.19
(–1.5, –0.5)	52.1	(–0.5, –0.5)	68.57	(0.5, –0.5)	77.98	(1.5, –0.5)	80.33
(–1.5, –1.5)	33.58	(–0.5, –1.5)	50.05	(0.5, –1.5)	59.46	(1.5, –1.5)	61.81

5.3.3 Univariate method

The univariate method generates trial solution for one decision variable keeping all others fixed. By this way, the best solution for each of the decision variables keeping others constant is obtained. The whole process is repeated iteratively until it converges. This is the easiest methods used to minimize functions of n variables where we find the minimum of the objective function by varying only one variable at a time, while holding all other variables fixed. Therefore, we are converting the process a one-dimensional minimization along each of the coordinate directions of an n-dimensional design space. This method is called the univariate search method.

Univariate method generates trial solution for one decision variable keeping all others fixed. Consider the ith iteration where we start at a base point X_i and vary that variable keeping the values of $n-1$ variables fixed. The problem turns into a one-dimensional minimization problem as we are changing only one variable. Therefore, we can use any of the methods discussed in Chapter 3 to generate a new base point X_{i+1}. Best solutions for each of the variables keeping others constant are obtained. Hereafter, the search is extended in a new direction. This search in new direction is accomplished by varying any one of the remaining $n-1$ variables that were fixed in the preceding iteration. In practice, this search process is continued by considering each coordinate direction one by one. The first cycle is completed after all the n directions are searched sequentially, and therefore, we repeat the whole process of sequential minimization [Rao, (2009)]. The whole process is repeated iteratively until convergence.

Theoretically, the univariate method is applicable for finding the minimum of any function that has continuous derivatives. However, the method may not even converge if the function possesses a steep valley. This process is very efficient for a quadratic function of the form

$$f(X)=\sum_{i=1}^{n} c_i x_i^2 \tag{5.7}$$

as the search directions line up with the principle axes as shown in Fig. 5.4a. However, this method does not work satisfactorily for more general form of quadratic objective functions as shown

$$f(X)=\sum_{i=1}^{n}\sum_{j=1}^{n} d_{ij} x_i x_j \tag{5.8}$$

For the later case as shown in Fig. (5.4b), the changes in X decreases gradually as it reaches close to the optimum point, so large number of iterations will be required to achieve high accuracy [Edgar *et al.* (2001)].

The univariate method is quite simple and can be performed easily. However, it does not converge quickly to an optimum solution because it has an oscillating tendency that progressively declines toward the optimum point. Therefore, it will be excellent if we stop the calculations at some point close to the optimum point instead of trying to get the exact optimum point [Rao, (2009)].

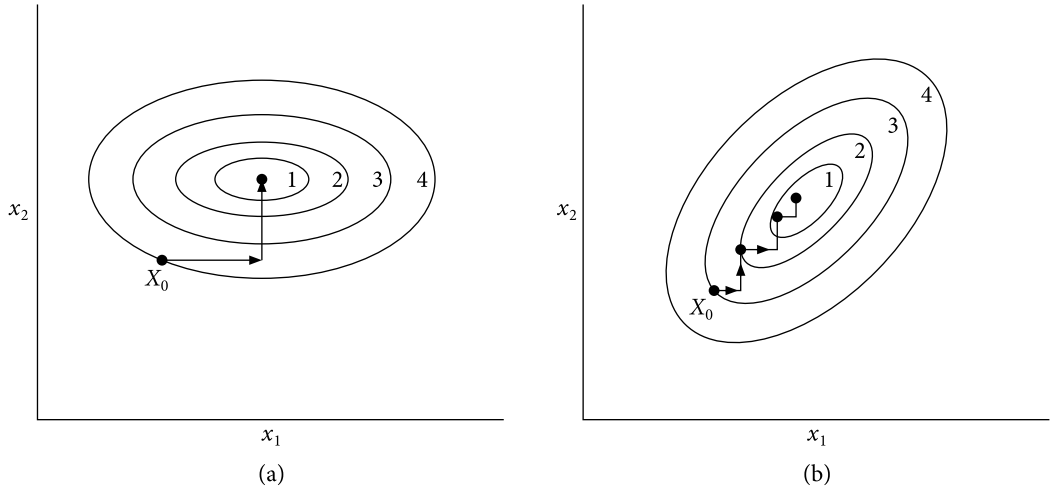

Fig. 5.4 Contour for quadratic function

5.3.4 Pattern search methods

In the univariate method, the search process is carried out for the minimum point along directions parallel to the coordinate axes. It is observed that this process might not converge in some circumstances and that even if the process converges, it will become very sluggish as it move toward the optimum point. We can avoid these difficulties by changing the search directions in a convenient way rather than keeping them constantly parallel to the coordinate axes. These changing directions are known as pattern directions. This process of going from a given point to the subsequent point is called '*move*'. A *move* is considered as success if the value of $f(X)$ decreases; otherwise, it is a

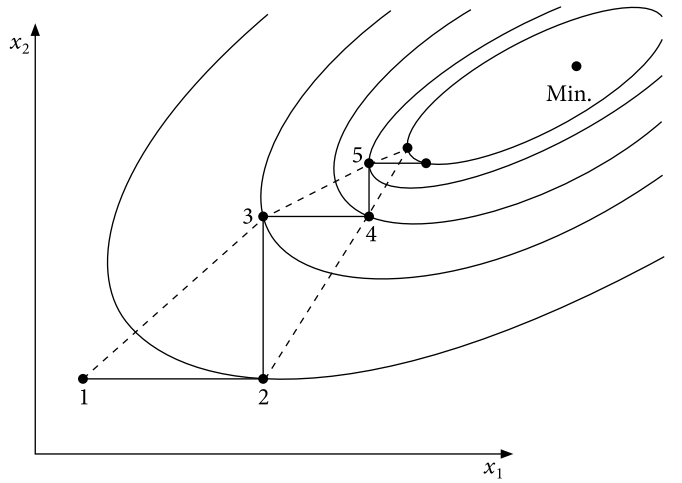

Fig. 5.5 Pattern search method

failure [Hooke and Jeeve (1960)]. It can be shown that for a quadratic objective function with two variables, all of these lines pass through a minimum. Unfortunately, this feature will not be true for functions with multiple variables, even if they are quadratics in nature. However, this concept can still be employed to accomplish a rapid convergence during the minimization of function with *n*-variable.

In Fig. 5.5, the move direction $1 \to 2 \to 3 \to 4 \to 5$ is univariate move whereas move direction $1 \to 3 \to 5$ (dotted line) is called pattern direction. Techniques that employ pattern directions as search directions are called the pattern search methods.

5.3.4.1 Powell's method

In the year 1964, Powell [Powell (1964)] delineated a method for finding out the minimizers without calculating derivatives. This method depends upon the characteristics of conjugate directions defined by a quadratic function. Powell's method can be explained as an extension of the original pattern search technique. This is one of the most extensively used direct search method and can be shown to be a method of conjugate directions. A conjugate directions method takes a finite number of steps to minimize a quadratic function. A conjugate directions method is supposed to accelerate the convergence of even general nonlinear objective functions because, near its minimum point, a general nonlinear function can be fairly well approximated by a quadratic function. The process of generating the conjugate directions, and the property of quadratic convergence are given in the following section.

To understand the Powell's pattern search method, we should first be well aware of the two important concepts, the conjugate directions and quadratic convergence.

A particular way of achieving quadratic termination is to call upon the idea of conjugacy of a set of non-zero vectors $s_1, s_2, \ldots, s_n$ to a specified positive definite matrix H. The property can be represented as

$$\left(s_i\right)^T H s_j = 0 \quad \forall i \neq j \tag{5.9}$$

A conjugate direction method is one that produces this kind of directions when it is employed to a quadratic function with Hessian H.

Theorem 5.1

A conjugate direction method terminates for a quadratic function in at most *n* exact line searches, and each X_{k+1} is the minimizer in the subspace generated by X_1 and the directions $s_1, s_2, \ldots, s_n$ (that is the set of points $\left\{X \middle| X = X_1 + \sum_{j=1}^{k} \alpha_j s_j \forall \alpha_j\right\}$).

The proof of this theorem is beyond the scope of this book. Readers may follow [Fletcher, 1987] for proof of this theorem.

Example 5.2

Consider the objective function:

$$f\left(X\right) = 2x_1^2 + x_2^2 - 5 \tag{5.10}$$

For minimizing the objective function, find the conjugate direction to the initial direction s_0. The starting point is $X_0 = [1 \quad 1]^T$ with the initial direction being $s_0 = [-4 \quad -2]^T$

Solution

The given values are $X_0 = [1 \quad 1]^T$ and $s_0 = [-4 \quad -2]^T$; therefore,

$$s_0 = -\begin{bmatrix} 4 \\ 2 \end{bmatrix} \text{ and } H_0 = \begin{bmatrix} 4 & 0 \\ 0 & 2 \end{bmatrix}$$

we have to solve Eq. (5.9) for $s_1 = \begin{bmatrix} s_1^1 & s_1^2 \end{bmatrix}^T$ with $s_0 = \begin{bmatrix} -4 & -2 \end{bmatrix}^T$. The equation is given below

$$\left(s_i\right)^T H s_j = 0 \quad 0 \le i \ne j \le n-1$$

substituting, the corresponding values we have

$$\begin{bmatrix} -4 & -2 \end{bmatrix} \begin{bmatrix} 4 & 0 \\ 0 & 2 \end{bmatrix} \begin{bmatrix} s_1^1 \\ s_1^2 \end{bmatrix} = 0$$

Because s_1^1 is not unique, we can choose $s_1^1 = 1$ and determine s_1^2 from above correlation

$$\begin{bmatrix} -16 & -4 \end{bmatrix} \begin{bmatrix} s_1^1 \\ s_1^2 \end{bmatrix} = 0$$

Thus, $s_1 = [1 \quad -4]^T$ is a direction conjugate to $s_0 = [-4 \quad -2]^T$.

We can achieve the minimum of $f(X)$ in two stages by utilizing first s_0 and then s_1.

5.3.4.2 Hooke–Jeeves method

Hooke–Jeeves method finds the minimum of a multivariable, unconstrained, nonlinear function. This is simple and fast optimizing method, which follows the 'hill climbing' technique. The algorithm takes a step in different 'directions' from the initial starting point, and performs a new model run. Then, the algorithm uses the new point as its best guess if the new likelihood score is better than the old one. When it is worse then, the algorithm remains unchanged to the old point. The search process continues in a series of these steps, each step slightly smaller than the previous one. When the algorithm finds a point, and any improvement is not possible with a small step in any direction, then this point is accepted as being the 'solution', and exits.

Hooke–Jeeves method consists of two main routines namely the "exploratory search" routine and the "pattern move" routine. The exploratory routine search the local proximity in the directions parallel to the coordinate axes for an improved objective function value. The pattern routine accelerates the search by moving to a new improved position in the direction of the previous optimal point obtained by the exploratory routine. A set of search direction is generated iteratively in the pattern search method. The main criterion for generating search directions is that they should traverse the search space completely. The search direction should be such that starting

from one point in the search space, we can reach any other point in the search space by traversing along these search directions only. For any N-dimensional problem, as a minimum N number of linearly independent search directions are required. For instance, at least two search directions are necessary to move from any one point to any other point in a two-variable function. There are many probable combinations of N search directions, a few of them may be able to attain the goal faster (with lesser number of iterations), and some of them may need more iterations. An exploratory move is conducted systematically in the neighborhood of the current point to get the best point around the current point. Subsequently, two such points are used to make a pattern move [Deb, (2009)].

Exploratory move:

The base point or the current solution is represented by X^c. Consider that the variable X_i^c is perturbed by Δ_i. Set $i = 1$ and $X = X^c$.

Step 1 Compute $f = f(X)$, $f^+ = f(X_i + \Delta_i)$ and $f^- = f(X_i - \Delta_i)$.

Step 2 Find $f_{\min} = \min(f, f^+, f^-)$. Then, set X corresponds to $f_{\min}$.

Step 3 If $i = N$ go to step 4; else set $i = i + 1$ and go to Step 1.

Step 4 If $X \neq X^c$, success; Else failure.

During the exploratory move, the current point is perturbed in both negative and positive directions along each variable one at a time and the best point is recorded. At the end of each variable perturbation, the current position is changed to the best point. When the point obtained at the end of all variable perturbations is different from the original point then the exploratory move is a successful one; otherwise, the exploratory move is a failure. In any circumstance, the best point is believed to be the result of the exploratory move.

Pattern move:

A new point is obtained by jumping from the current best point X^c along a direction connecting the previous best point $X^{(k-1)}$ and the current base point $X^{(k)}$ as follows:

$$X_p^{(k-1)} = X^{(k)} + \left(X^{(k)} - X^{(k-1)}\right) \tag{5.11}$$

The Hooke–Jeeves method involves an iterative application of an exploratory move in the vicinity of the current point and a subsequent jump using the pattern move. If the pattern move does not take the solution to a better region, the pattern move is not accepted and the extent of the exploratory search is reduced. The algorithm works as follows:

Step 1 Choose the following parameters: the starting point $X^{(0)}$, variable increments $\Delta_i (i = 1, 2, \ldots, N)$, a step reduction factor $\alpha > 1$, and a termination parameter, ε. Set $k = 0$.

Step 2 Perform an exploratory move considering $X^{(k)}$ as the base point. Say X is the result of the exploratory move. If the exploratory move is success, set $X^{(k+1)} = X$ and move to Step 4;

Else, move to Step 3.

Step 3 If $\|\Delta\| < \varepsilon$, Terminate;

Else, set $\Delta_i = \dfrac{\Delta_i}{\alpha}$ for $i = 1, 2, \ldots, N$ and go to Step 2.

Step 4 Set $k = k + 1$ and conduct the pattern move $X_p^{(k-1)} = X^{(k)} + \left(X^{(k)} - X^{(k-1)}\right)$.

Step 5 Conduct another exploratory move with $X_p^{(k-1)}$ as the base point. Let the result be $X^{(k-1)}$.
Step: Is $f\left(X^{(k+1)}\right) < f\left(X^{(k)}\right)$? If yes, go to Step 4;
Else go to Step 3.

The search strategy is simple and straightforward. The algorithm requires less storage for variables; only two points ($X^{(k)}$ and $X^{(k-1)}$) need to be stored at any iteration. The numerical calculations involved in the process are also simple. However, since the search largely depends on the moves along the coordinate directions (X_1, X_2, and so on) during the exploratory move, the algorithm may prematurely converge to a wrong solution, especially in the case of functions with highly nonlinear interactions among variables. The algorithm may also get stuck in the loop of generating exploratory moves either between Steps 5 and 6 or between Steps 2 and 3. Another feature of this algorithm is that it terminates only by exhaustively searching the vicinity of the converged point. This requires a large number of function evaluations for convergence to a solution with a reasonable degree of accuracy. The convergence to the optimum point depends on the parameter α. A value $\alpha = 2$ is recommended.

The pattern search method is clearly a simple strategy which is easily programmed and which require very little computer storage, and it has been found to be extremely useful in a wide variety of applications ranging from curve fitting to the on-line performance optimization of chemical processes [Swann, (1972)].

The working principle of the algorithm can be better understood through the following hand-simulation on a numerical exercise problem.

Example 5.3

Show the exploratory move for the function given below

$$\min F(X) = 1200\left(2x_1x_3 + 2x_2x_3\right) + 2500x_1x_2 + 500\left(\frac{1000}{x_1x_2x_3}\right) + 100\left(\frac{1000}{10x_1x_2x_3}\right) \qquad (5.12)$$

Solution

The value represented here is $f(X) = F(X)/1000$

Set iteration number $i = 1$

Consider the current point $X^{(1)} = [1 \;\; 1 \;\; 1]^T$ and $f^{(1)} = 517.3$

The increment vector $\Delta_1 = [0.2, 0.2, 0.2]^T$

now $f^+ = f(1.2, 1.2, 1.2) = 305.651$

and $f^- = f(0.8, 0.8, 0.8) = 1000.766$

$f_{\min} = \min(f^+, f, f^-) = 305.3$

Set $X^{(2)} = [1.2 \;\; 1.2 \;\; 1.2]^T$ for next iteration $i = 2$

Set iteration number $i = 2$

for this iteration $f = f(1.2, 1.2, 1.2) = 305.651$

Calculate $f^+ = f(1.4, 1.4, 1.4) = 200.167$

$f^- = f(1.0, 1.0, 1.0) = 517.3$

Therefore, $f_{\min} = (f^+, f, f^-) = 200.167$

Note For each iteration, we need to calculate the objective function value only at one point; other two values are stored in the previous iteration.

5.4 Gradient Search Method

Gradient search methods require the gradient vector of the objective function. Consider a function $f(X)$ where $X = [x_1, x_2, \ldots x_n]^T$. The gradient vector of $f(X)$ is given by the partial derivatives with respect to each of the independent variables

$$\nabla f(X) \equiv g(X) \equiv \left[\frac{\partial f}{\partial x_1} \quad \frac{\partial f}{\partial x_2} \quad \cdots \quad \frac{\partial f}{\partial x_n} \right]^T \tag{5.13}$$

The gradient possess a very useful property. In an n-dimensional space if we move from any point along the gradient direction, the value of $f(X)$ increases at the fastest rate. Therefore, the gradient direction is known as the direction of steepest ascent. Unfortunately, this steepest ascent direction is not a global property and it is a local one as the shape of the contours is not uniform.

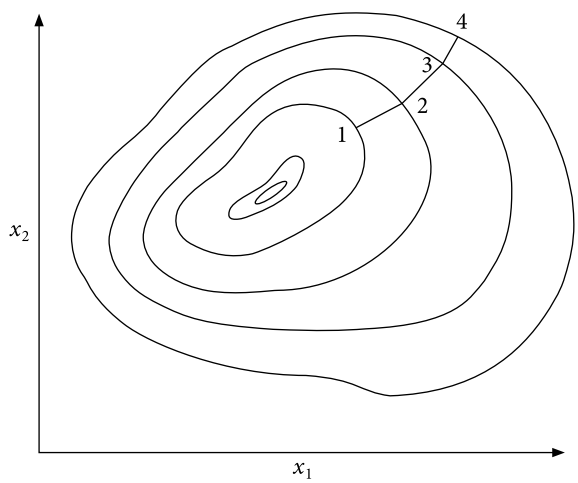

Fig. 5.6 Gradient search method

Figure 5.6 shows that the direction of search changes from point to point. At point 1, steepest direction is 1–2. Similarly, at point 2 and 3, steepest directions are 2–3 and 3–4 respectively.

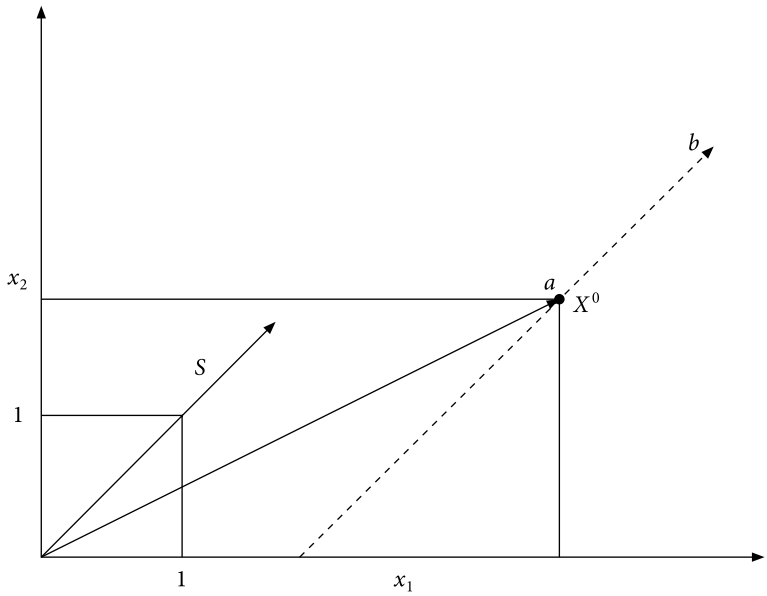

Fig. 5.7 Direction of movement of any point

Figure 5.7 explains how any point moves from a particular point X^0 with a direction s. Here, the line "$a - b$" starts from point a (X^0) and moves with a direction s = [1, 1].

5.4.1 Steepest descent (Cauchy) method

Cauchy in 1847 [Cauchy, 1847] first utilized the negative of the gradient vector ($-g(X)$) as a the direction for minimization problem. The gradient is the vector at any point X, which gives the direction (local) of the greatest rate of change of $f(X)$. It is orthogonal to the contour of $f(X)$ at X. Steepest descent method starts from an initial trial point X_1 and move iteratively along the directions of steepest descent until the optimum point is reached. The steepest descent method can be described by the following steps:

Let $f : \mathbb{R}^n \to \mathbb{R}$ be real-valued function of n variables:

1. Start from an arbitrary initial point X_1. Set the iteration number as $i = 1$.
2. Find the direction of search s_i as

$$s_i = -\nabla f_i = -\nabla f\left(X_i\right) \tag{5.14}$$

3. Calculate λ_i^*, the optimal step length in the direction s_i and set

$$X_{i+1} = X_i + \lambda_i^* s_i = X_i - \lambda_i^* \nabla f_i \tag{5.15}$$

here, λ_i^* determines how far to go in the given direction

4. Check the new point, X_{i+1}, for optimality. If X_{i+1} is optimum, stop the search process. If not, move to step 5.
5. Set the new iteration number $i = i + 1$ and go to step 2.

This process may seem to be the best unconstrained minimization method as all one-dimensional search start in the "best" direction. However, this process is not really effective in most problems as the steepest descent direction is a local property [Rao, (2009)]. Another drawback of this method is the convergence of this method largely depends on the nature of the objective function. The iterate can oscillate back and forth in a zigzag manner, making the process very slow.

Convergence Criteria:

The following criteria can be applied to terminate the iterative process.

1. If the change of the function value in two successive iterations is very small:

$$\left|\frac{f(X_{i+1}) - f(X_i)}{f(X_i)}\right| \leq \varepsilon_1 \tag{5.16}$$

2. If the values of the partial derivatives (components of the gradient) of the function f are very small:

$$\left|\frac{\partial f}{\partial X_i}\right| \leq \varepsilon_2 \qquad i = 1, 2, \ldots, n \tag{5.17}$$

3. If the change in the design vector in two successive iterations is very small:

$$\left|X_{i+1} - X_i\right| \leq \varepsilon_3 \tag{5.18}$$

Here, $\varepsilon_1, \varepsilon_2$ and ε_3 are very small numbers.

Example 5.4

Find the optimum of the following function using Cauchy's method.

$$f(X) = 9x_1^2 + 4x_1x_2 + 7x_2^2$$

Solution

The elements of the gradient are

$$\frac{\partial f}{\partial x_1} = 18x_1 + 4x_2 \text{ and } \frac{\partial f}{\partial x_2} = 4x_1 + 14x_2$$

Now we will employ the steepest descent method to get the solution

Consider $X_0 = [1 \quad 1]^T$, then

$f(X_0) = 20$ and

$\nabla f(X_0) = \begin{bmatrix} 22 & 18 \end{bmatrix}^T$

from Eq. (5.15), we have

$X_1 = X_0 - \lambda_0^* \nabla f(X_0)$

or $\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix} - \lambda_0^* \begin{bmatrix} 22 \\ 18 \end{bmatrix}$

now we have to find λ_0^* such that $f(X_1)$ will be minimum.

If we substitute value of X_1 in equation

$f(X) = 9x_1^2 + 4x_1x_2 + 7x_2^2$

we have $f(\lambda_0^*) = 20 - 808\lambda_0^* + 8208(\lambda_0^*)^2$

The optimum value of λ_0^* is 0.05 when $\frac{df(\lambda_0^*)}{d\lambda_0^*} = 0$

therefore, $X_1 = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix} - 0.05 \begin{bmatrix} 22 \\ 18 \end{bmatrix} = \begin{bmatrix} -0.1 \\ 0.1 \end{bmatrix}$ and $f(X_1) = 0.12$

we will reach the optimum point following this iterative method.

5.4.2 Conjugate gradient (Fletcher–Reeves) method

In view of Theorem 5.1, which equates conjugacy and exact line searches with quadratic termination, it is attractive to try to associate conjugacy properties with the steepest descent method in an attempt to achieve both efficiency and reliability. This is the aim of conjugate gradient method [R. Fletcher, (1987)].

The convergence characteristics of the steepest descent method can be greatly improved by modifying this into a conjugate gradient method. We can also consider this method as a conjugate directions method including the application of the gradient of the function [Rao, (2009)]. In the previous discussion, we have seen that a minimization method is quadratically convergent when it uses the conjugate directions. This characteristic of quadratic convergence is quite helpful since it gives assurance that this process will minimize a quadratic function in n steps or less. Every

quadratically convergent method is supposed to achieve the optimum point in a finite number of iterations because near the optimum point any general function can be well approximated by a quadratic function. We have noticed that Powell's conjugate direction method performs n single-variable minimizations per iteration and at the end of each iteration it establishes a new conjugate direction. Therefore, usually, n^2 single-variable minimizations are required to determine the minimum of a quadratic function. In contrast, if we are able to estimate the gradients of the objective function, we can establish a new conjugate direction after each one-dimensional minimization, and consequently a faster convergence can be achieved. The development of conjugate directions and formulation of the Fletcher–Reeves method are explained in the following section.

Development of the Fletcher–Reeves Method

The Fletcher–Reeves method is formulated by performing some modification of the steepest descent method in such a way that it becomes quadratically convergent. The search process starts from an arbitrary point X_1, the quadratic function

$$f(X) = \frac{1}{2} X^T [A] X + B^T X + C \tag{5.19}$$

can be minimized by using the search procedure along the steepest descent search direction $s_1 = -\nabla f_1$ with the step length

$$\lambda_1^* = -\frac{s_1^T \nabla f_1}{s_1^T A s_1} \tag{5.20}$$

A linear combination (Eq. (5.21)) of s_1 and $-\nabla f_2$ is used to find the direction of second search (s_2)

$$s_2 = -\nabla f_2 + \beta_2 s_1 \tag{5.21}$$

where the constant β_2 can be estimated by making s_1 and s_2 conjugate with respect to [A]. This lead to the following equation.

$$\beta_2 = -\frac{\nabla f_2^T \nabla f_2}{\nabla f_1^T s_1} = \frac{\nabla f_2^T \nabla f_2}{\nabla f_1^T \nabla f_1} \tag{5.22}$$

This process has been continued to derive the general formula for the ith search direction as

$$s_i = -\nabla f_i + \beta_i s_{i-1} \tag{5.23}$$

where

$$\beta_i = \frac{\nabla f_i^T \nabla f_i}{\nabla f_{i-1}^T \nabla f_{i-1}} \tag{5.24}$$

Therefore, the algorithm of Fletcher–Reeves method can be written as follows.

Algorithm

The algorithm of Fletcher–Reeves method is given below:

1. Start from X_1, an arbitrary initial point.
2. Calculate the first search direction $s_1 = -\nabla f(X_1) = -\nabla f_1$.
3. Find the point X_2 using the relation

$$X_2 = X_1 + \lambda_1^* s_1 \tag{5.25}$$

where λ_1^* is the optimal step length in the direction s_1. Set $i = 2$ and move to the next step.

4. Determine $\nabla f_i = \nabla f(X_i)$, and set

$$s_i = -\nabla f_i + \frac{|\nabla f_i|^2}{|\nabla f_{i-1}|^2} s_{i-1} \tag{5.26}$$

5. Calculate λ_i^*, the optimum step length in the direction s_i, and locate the new point

$$X_{i+1} = X_i + \lambda_i^* s_i \tag{5.27}$$

6. Check for the optimality of the point X_{i+1}. Stop the process if X_{i+1} is optimum; or else, set $i = i + 1$ and go to step 4.

Remark

This process should converge in n cycles or less for a quadratic function as the directions s_i used in this method are A-conjugate. However, the method may utilize much more than n cycles for convergence when quadratics are ill-conditioned (when contours are highly distorted and eccentric). The cumulative effect of rounding errors has been considered as the main reason for this. Since, s_i is represented by Eq. (5.26), any error emanating from the inaccuracies occurred during the estimation of λ_i^*, and from the round-off error in the process of the accumulation of successive $|\nabla f_i|^2 s_{i-1} / |\nabla f_{i-1}|^2$ terms, is carried forward through the vector s_i. Therefore, these errors progressively contaminate the search directions s_i. Thus in practice, it is required to restart the process periodically after some number of steps; say, m number of steps by considering the new search direction as the steepest descent direction. Therefore, after every m steps, s_{m+1} is replaced by $-\nabla f_{m+1}$ rather than using its usual form (Eq. (5.26)). A value of $m = n + 1$ is suggested by Fletcher and Reeves, where n denotes the number of design variables.

5.4.3 Newton's method

Newton's method can be used efficiently to minimize the multivariable functions. For this purpose, we have to consider a quadratic approximation of the function $f(X)$ at $X = X_i$ by means of the Taylor's series expansion

$$f(X) \approx f(X_i) + \nabla f_i^T (X - X_i) + \frac{1}{2}(X - X_i)^T [H_i](X - X_i) \tag{5.28}$$

where $[H_i] = [H]|_{X_i}$ is called the Hessian matrix (matrix of second partial derivatives) of f estimated at the point X_i. For the minimum of $f(X)$, the partial derivatives of Eq. (5.28) have been set equal to zero

$$\frac{\partial f(X)}{\partial X_j} = 0, \quad j = 1,2,\ldots,n \tag{5.29}$$

Form Eqs (5.28) and (5.29), we get

$$\nabla f = \nabla f_i + [H_i](X - X_i) = 0 \tag{5.30}$$

If $[H_i]$ is nonsingular, Eq. (5.30) can be solved to get an improved approximation ($X = X_{i+1}$) as

$$X_{i+1} = X_i - [H_i]^{-1} \nabla f_i \tag{5.31}$$

Since, the terms of higher-order have been neglected in Eq. (5.28). The Eq. (5.31) should be employed iteratively to get the optimum solution X^*. Provided that $[H_1]$ is nonsingular, it can be shown that the series of points $X_1, X_2, \ldots, X_{i+1}$ converge to the actual solution X^* starting from any initial point X_1 that is satisfactorily close to the solution X^*. It can be noticed that Newton's method is a second-order method since the second partial derivatives of the objective function is utilized in the form of the matrix $[H_i]$.

Example 5.5

Minimize the function using Newton's method

$$f(X) = 5x_1^2 + x_2^2 - 3x_1x_2$$

Starting point $X_0 = [1 \quad 1]^T$

Solution

Calculate the value of

$$\nabla f(X) = \begin{bmatrix} 10x_1 - 3x_2 \\ 2x_2 - 3x_1 \end{bmatrix} \quad \nabla f(X_0) = \begin{bmatrix} 7 \\ -1 \end{bmatrix}$$

and

$$H(X) = \begin{bmatrix} 10 & -3 \\ -3 & 2 \end{bmatrix} \quad H^{-1}(X) = \begin{bmatrix} 0.1818 & 0.2727 \\ 0.2727 & 0.9091 \end{bmatrix}$$

from Eq. (5.31)

$$X_1 = X_0 - \left[H_0\right]^{-1} \nabla f_0$$

$$X_1 = \begin{bmatrix} 1 & 1 \end{bmatrix}^T - \begin{bmatrix} 0.1818 & 0.2727 \\ 0.2727 & 0.9091 \end{bmatrix} \begin{bmatrix} 7 \\ -1 \end{bmatrix}$$

$$X_1 = \begin{bmatrix} 1 & 1 \end{bmatrix}^T - \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

$$X_1 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

calculate the value of the function at the point X_1

$$f\left(X_1\right) = f\left(X^*\right) = 0$$

Note The solution is found in a single step using Newton's method

5.4.4 Marquardt method

When the design vector X_i is away from the optimum point X^*, the steepest descent method adequately reduces the value of the function. On the other hand, the Newton method, converges fast when the design vector X_i is very close to the optimum point X^*. The Marquardt method [Marquardt, 1963] tries to take benefit of both the Newton's method and steepest descent. In this method, the diagonal elements of the Hessian matrix, $[H_i]$ have been modified by the equation

$$\left[\tilde{H}_i\right] = \left[H_i\right] + \alpha_i \left[I\right] \tag{5.32}$$

where $[I]$ and α_i represent an identity matrix and a positive constant respectively. The constant α_i ensures the positive definiteness of $\left[\tilde{H}_i\right]$ when $[H_i]$ is not positive definite. We can note that while α_i is sufficiently large (on the order of 10^4), the term $\alpha_i \left[I\right]$ dominates $[H_i]$ and the inverse of the matrix $\left[\tilde{H}_i\right]$ becomes

$$\left[\tilde{H}_i\right]^{-1} = \left[\left[H_i\right] + \alpha_i \left[I\right]\right]^{-1} \approx \left[\alpha_i \left[I\right]\right]^{-1} = \frac{1}{\alpha_i}\left[I\right] \tag{5.33}$$

Therefore, if the search direction s_i is calculated as

$$s_i = -\left[\tilde{H}_i\right]^{-1} \nabla f_i \tag{5.34}$$

For large values of α_i, the direction s_i converts to a direction of steepest descent. In the Marquardt method, the value of α_i is considered to be large during the starting and then decreased gradually to zero as the process proceeds iteratively. Therefore, the characteristics of this search technique change from those of a steepest descent method to those of the Newton method as the value of α_i decreases from a large value to zero. The iterative procedure of the modified version of Marquardt method can be illustrated as given bellow.

Algorithm

1. Start with the following parameters: X_1, an arbitrary initial point; a large constant α_1 (on the order of 10^4), a small constant ε (on the order of 10^{-2}), $c_1 (0 < c_1 < 1)$, and $c_2 (c_2 > 1)$. Set the iteration number as $i = 1$.
2. Calculate the gradient of the function, $\nabla f_i = \nabla f(X_i)$.
3. Check point X_i for optimality. If $\|\nabla f\|_i = \|\nabla f(X_i)\| \le \varepsilon$, X_i is optimum and thus stop the process. If not, move to step 4.
4. Estimate the new vector X_{i+1} by using the equation

$$X_{i+1} = X_i + s_i = X_i - \left[\left[H_i\right] + \alpha_i \left[I\right]\right]^{-1} \nabla f_i \tag{5.35}$$

5. Compare the values of f_{i+1} and f_i. If $f_{i+1} < f_i$, move to step 6. If $f_{i+1} \ge f_i$, go to step 7.
6. Set $\alpha_{i+1} = c_1 \alpha_i$, $i = i + 1$, and go to step 2.
7. Set $\alpha_i = c_2 \alpha_i$ and go to step 4.

The main advantage of this technique is the nonexistence of the step size λ_i along the search direction s_i. In fact, the above algorithm can be customized by introducing an optimal step length in Eq. (5.35) as

$$X_{i+1} = X_i + \lambda_i s_i = X_i - \lambda_i^* \left[\left[H_i\right] + \alpha_i \left[I\right]\right]^{-1} \nabla f_i \tag{5.36}$$

where λ_i^* is calculated using any of the one-dimensional search processes explained in Chapter 3.

Example 5.6

Minimize the function $f(X) = x_1 - x_2 + 3x_1^2 + 2x_1 x_2 + x_2^2$

Consider the starting point $X_1 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$ using Marquardt method with $\alpha_1 = 10^4$, $c_1 = \frac{1}{4}$, $c_2 = 2$ and $\varepsilon = 10^{-2}$

Solution

We will start with the first iteration

For $i = 1$:

The value of the function $f_1 = f(X_1) = 0.0$ and

$$\nabla f_1 = \begin{bmatrix} \dfrac{\partial f}{\partial x_1} \\ \dfrac{\partial f}{\partial x_2} \end{bmatrix}_{(0,0)} = \begin{bmatrix} 1+6x_1+2x_2 \\ -1+2x_1+2x_2 \end{bmatrix}_{(0,0)} = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

Since, $\|\nabla f_1\| = 1.4142 > \varepsilon$, we compute

$$[H_1] = \begin{bmatrix} \dfrac{\partial^2 f}{\partial x_1^2} & \dfrac{\partial^2 f}{\partial x_1 x_2} \\ \dfrac{\partial^2 f}{\partial x_1 x_2} & \dfrac{\partial^2 f}{\partial x_2^2} \end{bmatrix}_{(0,0)} = \begin{bmatrix} 6 & 2 \\ 2 & 2 \end{bmatrix}$$

$$X_2 = X_1 - \left[[H_1] + \alpha_1 [I]\right]^{-1} \nabla f_1$$

$$= \begin{bmatrix} 0 \\ 0 \end{bmatrix} - \begin{bmatrix} 6+10^4 & 2 \\ 2 & 2+10^4 \end{bmatrix}^{-1} \begin{bmatrix} 1 \\ -1 \end{bmatrix} = \begin{bmatrix} -0.9996 \\ 1.0000 \end{bmatrix} \times 10^{-4}$$

as $f_2 = f(X_2) = -1.9994 \times 10^{-4} < f_1$, we set $\alpha_2 = c_1 \alpha_1 = 2500$, $i = 2$, and proceed to the next iteration.

Iteration 2:

The gradient vector corresponding to X_2 is represented by

$$\nabla f_2 = \begin{bmatrix} \dfrac{\partial f}{\partial x_1} \\ \dfrac{\partial f}{\partial x_2} \end{bmatrix}_{(-0.9996\times10^{-4}, 1.0000\times10^{-4})} = \begin{bmatrix} 1+6x_1+2x_2 \\ -1+2x_1+2x_2 \end{bmatrix}_{(-0.9996\times10^{-4}, 1.0000\times10^{-4})} = \begin{bmatrix} 0.9996 \\ -1.0000 \end{bmatrix}$$

$\|\nabla f_2\| = 1.4139 > \varepsilon$, hence, we compute

$$X_3 = X_2 - \left[[H_2] + \alpha_2 [I]\right]^{-1} \nabla f_2$$

$$= \begin{bmatrix} -0.9996\times10^{-4} \\ 1.0000\times10^{-4} \end{bmatrix} - \begin{bmatrix} 2506 & 2 \\ 2 & 2502 \end{bmatrix}^{-1} \begin{bmatrix} 0.9996 \\ -1.0000 \end{bmatrix}$$

$$= \begin{bmatrix} -4.992 \times 10^{-4} \\ 5.000 \times 10^{-4} \end{bmatrix}$$

since, $f_3 = f(X_3) = -0.9987 \times 10^{-3} < f_2$, we set $\alpha_3 = c_1 \alpha_2 = 625$, $i = 3$, and proceed to the next iteration. The iteration process is to be continued until it satisfy the convergence criteria $\|\nabla f_i\| < \varepsilon$.

5.4.5 Quasi-Newton method

The Newton's method uses the basic iterative process that is given by Eq. (5.31):

$$X_{i+1} = X_i - [H_i]^{-1} \nabla f(X_i) \tag{5.37}$$

where $[H_i]$ is the Hessian matrix that consist of the second partial derivatives of the function f and varies with the design vector X_i for a non-quadratic (common nonlinear) objective function f. The fundamental concept used in the variable metric or Quasi-Newton methods is to approximate either $[H_i]$ by an another matrix $[A_i]$ or $[H_i]^{-1}$ by an another matrix $[B_i]$, considering only the first partial derivatives of f. If $[H_i]^{-1}$ is approximated by$[B_i]$, Eq. (5.37) can be represented as

$$X_{i+1} = X_i - \lambda_i^* [B_i] \nabla f(X_i) \tag{5.38}$$

where λ_i^* is considered as the optimal step length along the direction

$$s_i = -[B_i] \nabla f(X_i) \tag{5.39}$$

It can be observed that the steepest descent direction method can be shown as a special case of Eq. (5.39) by adjusting $[B_i] = [I]$.

Calculation of $[B_i]$

To execute the iteration process given by Eq. (5.38), an approximate inverse of the Hessian matrix, $[B_i] \equiv [A_i]^{-1}$, is to be calculated. For this purpose, first we expand the gradient of the function f about an arbitrary reference point (X_0) using Taylor's series as

$$\nabla f(X) \approx \nabla f(X_0) + [H_0](X - X_0) \tag{5.40}$$

If we select any two points X_i and X_{i+1} and use $[A_i]$ to approximate $[H_0]$, Eq. (5.40) can be written as

$$\nabla f_{i+1} = \nabla f(X_0) + [A_i](X_{i+1} - X_0) \tag{5.41}$$

$$\nabla f_i = \nabla f(X_0) + [A_i](X_i - X_0) \tag{5.42}$$

Subtracting Eq. (5.42) from Eq. (5.41) yields

$$g_i = [A_i] d_i \tag{5.43}$$

where

$$d_i = X_{i+1} - X_i \tag{5.44}$$

$$g_i = \nabla f_{i+1} - \nabla f_i \tag{5.45}$$

The solution of Eq. (5.43) for d_i can be given as

$$d_i = [B_i] g_i \tag{5.46}$$

where the inverse of the Hessian matrix, $[H_0]^{-1}$ is approximated by $[B_i] \equiv [A_i]^{-1}$. It can be noticed that Eq. (5.46) represents a system of n equations with n^2 unknown elements of the matrix $[B_i]$. So for n >1, the selection of $[B_i]$ is not unique and we have to select $[B_i]$ that is very close to $[H_0]^{-1}$, in some sense. Various number of procedures have been recommended in the literature for the calculation of $[B_i]$. Once $[B_i]$ is known, the value of $[B_{i+1}]$ is computed using the iterative process. The symmetry and positive definiteness of the matrix $[B_i]$ is to be maintained in addition to satisfying Eq. (5.46), which is a major concern for this method. If $[B_i]$ is symmetric and positive definite, the $[B_{i+1}]$ must remain symmetric and positive definite.

Rank 1 Updates

The matrix $[B_i]$ can be updated by using the general formula given as

$$[B_{i+1}] = [B_i] + [\Delta B_i] \tag{5.47}$$

where $[\Delta B_i]$ is considered as the correction or update matrix added to $[B_i]$. Theoretically it can be considered that the matrix $[B_i]$ have the rank as high as n. However, in practical application, most updates $[\Delta B_i]$ are only of either rank 1 or rank 2. To develop a rank 1 update, simply we select a scaled outer product of a vector z for $[\Delta B_i]$ as

$$[\Delta B_i] = czz^T \tag{5.48}$$

where the n-component vector z and constant c the are to be estimated. Combination of Eqs (5.47) and (5.48) gives us

$$[B_{i+1}] = [B_i] + czz^T \tag{5.49}$$

If Eq. (5.49) is forced to satisfy the Quasi-Newton condition, we have

$$d_i = [B_{i+1}] g_i \tag{5.50}$$

we get

$$d_i = \left(\left[B_i\right] + czz^T\right) g_i = \left[B_i\right] g_i + cz\left(z^T g_i\right) \tag{5.51}$$

since, $\left(z^T g_i\right)$ in Eq. (5.51) is a scalar, Eq. (5.51) can rewritten as

$$cz = \frac{d_i - \left[B_i\right] g_i}{z^T g_i} \tag{5.52}$$

Therefore, an easy selection for z and c would be

$$z = d_i - \left[B_i\right] g_i \tag{5.53}$$

$$c = \frac{1}{z^T g_i} \tag{5.54}$$

This gives us the unique rank 1 update formula for $[B_{i+1}]$:

$$\left[B_{i+1}\right] = \left[B_i\right] + \left[\Delta B_i\right] \equiv \left[B_i\right] + \frac{\left(d_i - \left[B_i\right] g_i\right)\left(d_i - \left[B_i\right] g_i\right)^T}{\left(d_i - \left[B_i\right] g_i\right)^T g_i} \tag{5.55}$$

This formula has been credited to Broyden [Broyden, (1967)]. To employ Eq. (5.55), at the start of the algorithm an initial symmetric positive definite matrix is selected for $[B_i]$. After that, the next point X_2 is calculateed using Eq. (5.38). Then the new matrix $[B_2]$ is calculated using Eq. (5.55) and the new point X_3 is found out from Eq. (5.38). Until the convergence is achieved, this iterative process is continued. Equation (5.55) confirms that whenever $[B_i]$ is symmetric, $[B_{i+1}]$ is also symmetric. Though, there is no assurance that $[B_{i+1}]$ will remain positive definite even if $[B_i]$ is positive definite. This may cause a breakdown of the method, particularly when it is applied for the optimization of non-quadratic functions. It can be confirmed easily that the columns of the matrix $[\Delta B_i]$ described by Eq. (5.55) are multiples of each other. The updating matrix possesses only one independent column and therefore, the rank of the matrix will be 1. For this reason, the Eq. (5.55) is considered to be a rank 1 updating formula. Although the Broyden formula given in Eq. (5.55) is not robust, it has the characteristic of quadratic convergence [Broyden *et al.* (1975)]. The rank 2 update formulas discussed in the next section give the assurance of both symmetry and positive definiteness of the matrix $[B_{i+1}]$. These formulas are more robust in minimizing common nonlinear functions; therefore, they are favored in practical applications.

Theorem 5.2

If it is well defined, and if $d_1, d_2, \ldots, d_n$ are independent, then the rank 1 method terminates on a quadratic function in at most $n + 1$ searches, with $B_i + 1 = H^{-1}$.

Rank 2 Updates

In rank 2 updates we decide the update matrix $[\Delta B_i]$ as the sum of two rank 1 updates as

$$[\Delta B_i] = c_1 z_1 z_1^T + c_2 z_2 z_2^T \tag{5.56}$$

where c_1 and c_2 are constants and z_1 and z_2 are the n-component vectors. We have to determine these constants and vectors. Combining Eqs (5.47) and (5.56) we get

$$[B_{i+1}] = [B_i] + c_1 z_1 z_1^T + c_2 z_2 z_2^T \tag{5.57}$$

By Eq. (5.57) is forced to satisfy the Quasi-Newton condition, Eq. (5.50), we get

$$d_i = [B_i] g_i + c_1 z_1 \left(z_1^T g_i\right) + c_2 z_2 \left(z_2^T g_i\right) \tag{5.58}$$

where $\left(z_1^T g_i\right)$ and $\left(z_2^T g_i\right)$ can be identified as scalars. Although the vectors z_1 and z_2 in Eq. (5.58) are not unique, the choices can be made to satisfy Eq. (5.58) are as follows:

$$z_1 = d_i \tag{5.59}$$

$$z_2 = [B_i] g_i \tag{5.60}$$

$$c_1 = \frac{1}{z_1^T g_i} \tag{5.61}$$

$$c_2 = \frac{1}{z_2^T g_i} \tag{5.62}$$

Therefore, the formula for rank 2 update can be stated as

$$[B_{i+1}] = [B_i] + [\Delta B_i] \equiv [B_i] + \frac{d_i d_i^T}{d_i^T g_i} - \frac{\left([B_i] g_i\right)\left([B_i] g_i\right)^T}{\left([B_i] g_i\right)^T g_i} \tag{5.63}$$

This equation is known as the Davidon–Fletcher–Powell (DFP) formula [Davidon, (1959)] [Fletcher and Powell, (1963)]. Since,

$$X_{i+1} = X_i + \lambda_i^* s_i \tag{5.64}$$

where s_i is the search direction, we can rewrite $d_i = X_{i+1} - X_i$ as

$$d_i = \lambda_i^* s_i \tag{5.65}$$

Thus, Eq. (5.63) can be represented as

$$[B_{i+1}] = [B_i] + \frac{\lambda_i^* s_i s_i^T}{s_i^T g_i} - \frac{([B_i] g_i)([B_i] g_i)^T}{([B_i] g_i)^T g_i} \tag{5.66}$$

Equation (5.66) gives the updated value of matrix B.

5.4.6 Broydon–Fletcher–Goldfrab–Shanno method

As discussed previously, the major difference of the BFGS and the DFP methods is that in the BFGS method, the Hessian matrix is updated iteratively instead of the inverse of the Hessian matrix. The following steps can illustrate the BFGS method.

1. Start with X_1, an initial point and a $n \times n$ positive definite symmetric matrix $[B_i]$ as an initial estimation of the inverse Hessian matrix of the function f. We can take $[B_1]$ as the identity matrix $[I]$ without any additional information. Calculate the gradient vector $\nabla f_1 = \nabla f(X_1)$ and set the iteration number as $i = 1$.
2. Calculate ∇f_i, the gradient of the function f at point X_i; and set

$$s_i = -[B_i] \nabla f_i \tag{5.67}$$

3. Find, λ_i^*, the optimal step length in the direction s_i and compute

$$X_{i+1} = X_i + \lambda_i^* s_i \tag{5.68}$$

4. Check the point X_{i+1} for optimality. If $\|\nabla f_{i+1}\| \le \varepsilon$, where the value of ε is very small, take $X^* \approx X_{i+1}$ and stop the process. Or else, move to step 5.
5. Update the Hessian matrix as

$$[B_{i+1}] = [B_i] + \left(1 + \frac{g_i^T [B_i] g_i}{d_i^T g_i}\right) \frac{d_i d_i^T}{d_i^T g_i} - \frac{d_i g_i^T [B_i]}{d_i^T g_i} - \frac{[B_i] g_i d_i^T}{d_i^T g_i} \tag{5.69}$$

where

$$d_i = X_{i+1} - X_i = \lambda_i^* s_i \tag{5.70}$$

$$g_i = \nabla f_{i+1} - \nabla f_i = \nabla f\left(X_{i+1}\right) - \nabla f\left(X_i\right) \tag{5.71}$$

6. Set the new iteration number as $i = i + 1$ and go to step 2.

Notes

1. We can consider the BFGS as a conjugate gradient, Quasi-Newton, and variable metric method.
2. The BFGS method can be termed an indirect update method, because the inverse of the Hessian matrix is approximated.
3. The matrix $[B_i]$ maintains its positive definiteness as the value of i increases, if the step lengths λ_i^* are found accurately. Therefore, for real life application, the matrix $[B_i]$ might become indefinite or even singular if λ_i^* are not computed correctly. For this reason, a resetting of the matrix $[B_i]$ to the identity matrix $[I]$ is required periodically. However, practical experience shows that the BFGS method is less sensitive to the errors in λ_i^* compared to the DFP method.
4. It has been found that the BFGS method has super-linear convergence near the point X^* [Dennis and More, (1977)].

5.5 Levenberg–Marquardt Algorithm

Newton's method with a line search algorithm converges quickly when $f(X)$ is convex. The matrix $H(X)$ may not be positive-definite in all places if $f(X)$ is not strictly convex (often it is the case in regions far from the optimum). Therefore, replacing $H(X)$ by another positive-definite matrix is one way to forcing convergence. The Marquardt–Levenberg method is one approach to accomplish this, as illustrated in the subsequent section.

Whenever the Hessian matrix $H(X_k)$ is not positive, the search direction $d_k = H(X_k)^{-1} g_k$ may not point in a descent direction. A simple procedure is followed to make sure that the search direction is a descent direction. The so-called Levenberg–Marquardt modification is introduced to Newton algorithm:

$$X_{k+1} = X_k - \left(H\left(X_k\right) + \mu_k I\right)^{-1} g_k \tag{5.72}$$

where $\mu \geq 0$

The fundamental concept of the Levenberg–Marquardt modification is given in this section. Consider H, a symmetric matrix that may not be positive definite. Let the eigenvalues of H are $\lambda_1, \ldots, \lambda_n$ and the corresponding eigenvectors are $v_1, \ldots, v_n$. The eigenvalues $\lambda_1, \ldots, \lambda_n$ are real, but may not all be positive. After that, consider the matrix $G = H + \mu I$, where $\mu \geq 0$. Note that the eigenvalues of G are $\lambda_1 + \mu, \ldots, \lambda_n + \mu$. Indeed,

$$Gv_i = \left(H + \mu I\right) v_i \tag{5.73}$$

$$= Hv_i + \mu I v_i \tag{5.73a}$$

$$= \lambda_i v_i + \mu v_i \tag{5.73b}$$

$$= (\lambda_i + \mu) v_i \tag{5.73c}$$

which shows that v_i is also an eigenvector of G with eigenvalue $\lambda_i + \mu$, for all $i = 1,\ldots, n$. Therefore, when the value of μ is sufficiently large, all the eigenvalues of G are positive, and G is positive definite. Consequently, if the parameter μ_k is sufficiently large in the Levenberg–Marquardt modification of Newton's algorithm, then the search direction $d_k = -\left(H(X_k) + \mu_k I\right)^{-1} g_k$ always points in a descent direction. In this case, if we further introduce a step size α_k as illustrated in the preceding section,

$$X_{k+1} = X_k - \alpha_k \left(H(X_k) + \mu_k I\right)^{-1} g_k \tag{5.74}$$

then it is assured that the descent property holds.

When we consider that $\mu_k \to 0$, the Levenberg–Marquardt modification of Newton's algorithm can be made to move toward the performance of the pure Newton's method. Conversely, by considering $\mu_k \to \infty$, the algorithm becomes a pure gradient method with small step size. In practice, we can start the process with a small μ_k, and then increase the value slowly till we observe that the iteration is descent, that is, $f(X_{k+1}) < f(X_k)$.

Levenberg–Marquardt method, first derived by Levenberg in 1944 [Levenberg, (1944)] and re-derived by Marquardt in 1963 [Marquardt, (1963)], is a method for solving nonlinear equations. This method is often mentioned when the history of trust region algorithms is discussed. The reason is that the technique of trust region is, in some sense, equivalent to that of the Levenberg–Marquardt method.

Consider a system of nonlinear equations

$$f_i(X) = 0\,,\ i = 1,2,\ldots,m \tag{5.75}$$

where $f_i(X)$, $i = 1, 2,\ldots, m$ are continuous differentiable functions in $\Re^n$. We try to compute a least squre solution, which means that we need to solve the nonlinear least squares problem

$$\min_{X \in \Re^n} \left\| F(X) \right\|_2^2 \tag{5.76}$$

where $F(X) = \left(f_1(X),\ldots, f_m(X)\right)^T$. The Gauss–Newton method is iterative, and at current iterate X_k, the Gauss-Newton step is

$$d_k = -\left(A(X_k)^T\right) + F(X_k) \tag{5.76}$$

Summary

- This chapter elucidates various techniques for optimization of multivariable unconstrained problem. Various search methods like direct search method (e.g., random search methods, grid search method, univariate method, and pattern search methods) and gradient search method (e.g., steepest descent (Cauchy) method, conjugate gradient (Fletcher-Reeves) method, Newton's method, Marquardt method, Quasi-Newton method, and BFGS method) has been discussed with examples. A modified form of Marquardt method that is Levenberg–Marquardt algorithm method is also discussed in this chapter.

Review Questions

5.1 Show that one iteration is sufficient to find the minimum of a quadratic function by using the Newton's method.

5.2 Give some examples of unconstrained optimization in the field of chemical engineering.

5.3 Find the of the function

$$f(X) = 2x_1^2 + x_2^2 - 2x_1x_2$$

using Random Jumping method.

5.4 The dye removal by TiO_2 adsorption (at pH 5.5) is given by (Anupam K. *et al.* 2011)

$$\text{dye removal} = 13.08(\text{TiO}_2) + 15.77(\text{Time}) - 9.996(\text{TiO}_2)^2 - 12.347(\text{Time})^2$$

Plot the contour and find the maximum dye removal using Grid Search method.

5.5 What are basic differences between Exploratory move and Pattern move?

5.6 Consider the following minimization problem

$$f(X) = x_1^2 + 2x_1x_2 + x_2^2 + 3x_1$$

Find the minimum using Steepest Descent method.

5.7 Solve the problem 6 using Newton's method.

5.8 What are advantages of Marquardt method over Steepest Descent and Newton's method.

5.9 The BFGS method can be considered as a Quasi-Newton, conjugate gradient, and variable metric method- Justify this statement with proper example.

5.10 In Quasi-Newton method, the matrix $[B_i]$ is updated using the formula

$$[B_{i+1}] = [B_i] + \frac{\lambda_i^* s_i s_i^T}{s_i^T g_i} - \frac{([B_i] g_i)([B_i] g_i)^T}{([B_i] g_i)^T g_i}$$

Discuss the effect of λ_i^* in this iteration process.

References

Anderson, R. L. *Recent Advances in Finding Best Operating Conditions*, J. Amer. Statist. Assoc., 48(1953): 789–98.

Anupam, K. *et al. The Canadian Journal of Chemical Engineering*, 89(2011): 1274–80.

Broyden, C. G. 1967. *Quasi-Newton Methods and their Application to Function Minimization,* Mathematics of Computation, vol. 21, pp. 368.

Broyden, C. G., Dennis, J. E. and More, J. J. *On the Local and Superlinear Convergence of Quasi-Newton Methods,* Journal of the Institute of Mathematics and Its Applications, 12(1975): 223.

Cauchy, A. L. 1847. *Méthode Générale Pour La Résolution Des Systèmes D'équations Simultanées,* Comptes Rendus de l'Academie des Sciences, Paris, vol. 25, pp. 536–38.

Davidon, W. C. 1959. *Variable Metric Method of Minimization,* Report ANL–5990, Argonne National Laboratory, Argonne, IL.

Deb, K. 2009. *Optimization for Engineering Design: Algorithms and Examples,* PHI Learning Pvt. Ltd.

Dennis, J. E. Jr. and More, J. J. 1977. *Quasi-Newton Methods, Motivation and Theory,* SIAM Review, vol. 19, No. 1, pp. 46–89.

Dutta, S. *et al. Adsorptive Removal of Chromium (VI) from Aqueous Solution Over Powdered Activated Carbon:* Optimisation through Response Surface Methodology, Chemical Engineering Journal, 173(2011): 135–43.

Edgar, T. F., Himmelblau, D. M., Lasdon, L. S. 2001. *Optimization of Chemical Processes,* McGraw-Hill.

Fletcher, R. and Powell, M. J. D. 1963. A Rapidly Convergent Descent Method for Minimization, Computer Journal, vol. 6, No. 2, pp. 163–68.

Fox, R. L. 1971. *Optimization Methods for Engineering Design,* Addison–Wesley, Reading, MA.

Hooke, R., Jeeves, T. A. *Direct Search Solution of Numerical and Statistical Problems,* Assoc. Computing Mechinery J., 8(1960): 212–29.

Hooke, R. and Jeeves T. A. 1961. *"Direct Search" Solution of Numerical and Statistical Problems*, J. Ass. Comput. Mach., 8: 212–29.

Karnopp, D. C., *Random Search Techniques for Optimization Problems*, Automatica, 1(1963): 111–21.

Levenberg, K. *"A Method for the Solution of Certain Nonlinear Problems in Least Squares",* Qart. Appl. Math., 2(1944): 164–66.

Lewis, R. M., Torczon, V., Trosset, M. W. *Direct Search Methods: Then and Now;* Journal of Computational and Applied Mathematics, 124(2000): 191–207.

Marquardt, D. W. *An Algorithm for Least-Squares Estimation of Nonlinear Inequalities,* SIAM J. Appl. Math., 11(1963): 431–41.

Montgomery, D. C. 2008. *Design and Analysis of Experiments*, John Wiley and Sons, INC.

Powell, M. J. D. An Efficient Method for Finding the Minimum of a Function of Several Variables without Calculating Derivatives, The Computer Journal, 7(1964): pp. 155–62.

Rao, S. S. 2009. *Engineering Optimization: Theory and Practice* (4*e*), John Wiley and Sons.

Rastrigin, L. A. The *Convergence of the Random Search Method in the External Control of a Many-Parameter System,* Automat. Remote Control, 24(1963): 1337–42.

Swann, W. H. 1972. *Direct Search Methods, in: W. Murray (Ed.), Numerical Methods for Unconstrained Optimization,* New York: Academic Press, pp. 13–28.